Search Results

Search Results for: **[breakpoint and watchpoint]**
Found **26** of **127,132 searched.**

Search within Results

Results 1 - 20 of 26 **short listing**

Prev Page 1 2 Next Page

1 Poor man's watchpoints 97%
Max Copperman , Jeff Thomas
ACM SIGPLAN Notices January 1995
Volume 30 Issue 1
Bugs that result from corruption of program data can be very difficult to track down without specialized help from a debugger. If the debugger cannot help the user find the point at which data gets corrupted, the user may have a long iterative debugging task. If the debugger is able to stop execution of the program at the point where data gets corrupted, as with watchpoints (also known as data breakpoints), it may be a very simple task to find a data corruption bug. In this paper, we discuss a m ...

2 Hardware support for program debuggers in a paged virtual memory 96%
David Abramson , John Rosenberg
ACM SIGARCH Computer Architecture News June 1983
Volume 11 Issue 2

3 Getting to Know gdb 87%
Mike Loukides , Andy Oram
Linux Journal September 1996

4 Fast hardware-software coverification by optimistic execution of real processor 85%
Sungjoo Yoo , Jong-Eun Lee , Jinyong Jung , Kyungseok Rha , Youngchul Cho , Kiyoung Choi
Pr ceedings f the c nference on Design, automati n and test in Europe January 2000

5 Tera hardware-software cooperation 85%

Gail Alverson , Preston Briggs , Susan Coatney , Simon Kahan , Richard Korry
Pr ceedings of the 1997 ACM/IEEE c nference on Supercomputing (CDROM)
November 1997

The development of Tera's MTA system was unusual. It respected the need for fast hardware and large shared memory, facilitating execution of the most demanding parallel application programs. But at the same time, it met the need for a clean machine model enabling calculated compiler optimizations and easy programming; and the need for novel architectural features necessary to support fast parallel system software. From its inception, system and application needs have molded the MTA architecture. ...

6 A software instruction counter 82%

J. M. Mellor-Crummey , T. J. LeBlanc
ACM SIGARCH Computer Architecture News , Proceedings of the third international conference on Architectural support for programming languages and operating systems April 1989
Volume 17 Issue 2

Although several recent papers have proposed architectural support for program debugging and profiling, most processors do not yet provide even basic facilities, such as an instruction counter. As a result, system developers have been forced to invent software solutions. This paper describes our implementation of a software instruction counter for program debugging. We show that an instruction counter can be reasonably implemented in software, often with less than 10% execution overhead. Ou ...

7 The p2d2 project: building a portable distributed debugger 82%

Robert Hood
Proceedings of the SIGMETRICS symposium on Parallel and distributed tools
January 1996

8 Recompilation for debugging support in a JIT-compiler 80%

Mustafa M. Tikir , Jeffrey K. Hollingsworth , Guei-Yuan Lueh
ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering November 2002
Volume 28 Issue 1

A static Java compiler converts Java source code into a verifiably secure and compact architecture-neutral intermediate format, called Java *byte codes*. The Java byte codes can be either interpreted by a Java Virtual Machine or translated into native code by Java Just-In-Time compilers. Static Java compilers embed debug information in the Java class files to be used by the source level debuggers. However, the debug information is generated for architecture independent byte codes and most o ...

9 Embra: fast and flexible machine simulation 80%

Emmett Witchel , Mendel Rosenblum
ACM SIGMETRICS Performance Evaluati n Review , Pr ceedings f the 1996 ACM SIGMETRICS international conference on Measurement and modeling of c mputer systems May 1996
Volume 24 Issue 1

This paper describes Embra, a simulator for the processors, caches, and memory systems of uniprocessors and cache-coherent multiprocessors. When running as part

of the SimOS simulation environment, Embra models the processors of a MIPS R3000/R4000 machine faithfully enough to run a commercial operating system and arbitrary user applications. To achieve high simulation speed, Embra uses dynamic binary translation to generate code sequences which simulate the workload. It is the first machine simu ...

10 Hardware and software support for efficient exception handling 80%

Chandramohan A. Thekkath , Henry M. Levy

Proceedings of the sixth international conference on Architectural support for programming languages and operating systems November 1994

Volume 29 , 28 Issue 11 , 5

Program-synchronous exceptions, for example, breakpoints, watchpoints, illegal opcodes, and memory access violations, provide information about exceptional conditions, interrupting the program and vectoring to an operating system handler. Over the last decade, however, programs and run-time systems have increasingly employed these mechanisms as a performance optimization to detect normal and expected conditions. Unfortunately, current archi ...

11 Features: Code Spelunking: Exploring Cavernous Code Bases 80%

George V. Neville-Neil

Queue September 2003

Volume 1 Issue 6

Try to remember your first day at your first software job. Do you recall what you were asked to do, after the human resources people were done with you? Were you asked to write a piece of fresh code? Probably not. It is far more likely that you were asked to fix a bug, or several, and to try to understand a large, poorly documented collection of source code.

Of course, this doesn't just happen to new graduates; it happens to all of us whenever we start a new job or look at a ...

12 Mondrian memory protection 80%

Emmett Witchel , Josh Cates , Krste Asanović

Tenth international conference on architectural support for programming languages and operating systems on Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X) October 2002

Volume 37 , 30 , 36 Issue 10 , 5 , 5

Mondrian memory protection (MMP) is a fine-grained protection scheme that allows multiple protection domains to flexibly share memory and export protected services. In contrast to earlier page-based systems, MMP allows arbitrary permissions control at the granularity of individual words. We use a compressed permissions table to reduce space overheads and employ two levels of permissions caching to reduce run-time overheads. The protection tables in our implementation add less than 9% overhead to ...

13 Dynamic currency determination in optimized programs 80%

D. M. Dhamdhere , K. V. Sankaranarayanan

ACM Transacti ns on Pr gramming Languages and Systems (TOPLAS) November 1998

Volume 20 Issue 6

Compiler optimizations pose many problems to source-level debugging of an optimized program due to reordering, insertion, and deletion of code. On such problem is to determine whether the value of a varible is current at a breakpoint—that is, whether

its actual value is the same as its expected value. We use the notion of dynamic currency of a variable in source-level debugging and propose the use of a minimal unrolled graph to reduce ...

14 Debugging optimized code with dynamic deoptimization 80%

Urs Hölzle , Craig Chambers , David Ungar

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation July 1992

Volume 27 Issue 7

SELF's debugging system provides complete source-level debugging (expected behavior) with globally optimized code. It shields the debugger from optimizations performed by the compiler by dynamically deoptimizing code on demand. Deoptimization only affects the procedure activations that are actively being debugged; all other code runs at full speed. Deoptimization requires the compiler to supply debugging information at discrete interrupt points

15 ReEnact: using thread-level speculation mechanisms to debug data races in multithreaded codes 77%

Milos Prvulovic , Josep Torrellas

ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture May 2003

Volume 31 Issue 2

While removing software bugs consumes vast amounts of human time, hardware support for debugging in modern computers remains rudimentary. Fortunately, we show that mechanisms for Thread-Level Speculation (TLS) can be reused to boost debugging productivity. Most notably, TLS's rollback capabilities can be extended to support rolling back recent buggy execution and repeating it as many times as necessary until the bug is fully characterized. These incremental re-executions are deterministic even i ...

16 The suitability of the VAX for a course in assembly language 77%

Robert W. Sebesta

ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE technical symposium on Computer science education February 1983

Volume 15 Issue 1

This paper describes the assembly language course we teach, using a Digital Equipment Corporation VAX-11/780 minicomputer, in which structured programming is stressed. It also discusses the relative merits and disadvantages of choosing the VAX as the computer to be used in such a course. The first section of the paper provides a quick survey of the VAX architecture. The second describes our course in assembly language, including our method of structuring assembly language program ...

17 Standardization approach of ITRON debugging interface specification and evaluation of its adaptability 77%

Takayuki Wakabayashi , Hiroaki Takada

ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and to ls for embedded systems: s ftware and compilers f r embedded systems June 2002

Volume 37 Issue 7

Debugging environments for embedded systems unavoidably depend on the internal structure of the operating system (OS) in order to implement OS support functions. Since the ITRON specification standardizes only the API, the internal structure of

operating systems conforming to the ITRON Specification are different, resulting in difficulties in supporting ITRON-Specification operating systems for debugging environments. To solve this problem, we design the ITRON Debugging Interface Specification w ...

18 Cheap hardware support for software debugging and profiling 77%

T. A. Cargill , B. N. Locanthi

Proceedings of the second international conference on Architectual support for programming languages and operating systems October 1987

Volume 15 , 22 , 21 Issue 5 , 10 , 4

19 A new framework for debugging globally optimized code 77%

Le-Chun Wu , Rajiv Mirani , Harish Patil , Bruce Olsen , Wen-mei W. Hwu

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation May 1999

Volume 34 Issue 5

With an increasing number of executable binaries generated by optimizing compilers today, providing a clear and correct source-level debugger for programmers to debug optimized code has become a necessity. In this paper, a new framework for debugging globally optimized code is proposed. This framework consists of a new code location mapping scheme, a data location tracking scheme, and an emulation-based forward recovery model. By taking over the control early and emulating instructions selective ...

20 Reverse execution of programs 77%

Bitan Biswas , R. Mall

ACM SIGPLAN Notices April 1999

Volume 34 Issue 4

Conventional debuggers do not allow users to go back and examine the program states at statements which have already been executed. In case the user wants to examine the program state at a statement which was executed sometime back, he is forced to restart the entire debugging process. To overcome this problem, we examine the issue of reverse execution of programs. To this end, we introduce the concept of *inverse of a statement*. We describe our implementation of a debugger which can execu ...

h c g e cf c